UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/069,805	11/07/2002	Gilbert Wolrich	10559-307US1	7118

20985 7590 05/22/2007
FISH & RICHARDSON, PC
P.O. BOX 1022
MINNEAPOLIS, MN 55440-1022

EXAMINER	
HUISMAN, DAVID J	
ART UNIT	**PAPER NUMBER**
2183	

MAIL DATE	DELIVERY MODE
05/22/2007	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Office Action Summary	Application No.	Applicant(s)	
	10/069,805	WOLRICH ET AL.	
	Examiner	Art Unit	
	David J. Huisman	2183	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *20 March 2007*.

2a)☒ This action is **FINAL**. 2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1,2 and 4-15* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1,2 and 4-15* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on *14 September 2006* is/are: a)☒ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☐ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☒ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date *2/22/2007 & 5/1/2007*.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

DETAILED ACTION

1. Claims 1-2, and 4-15 have been examined.

Papers Submitted

2. It is hereby acknowledged that the following papers have been received and placed of record in the file: IDS and Change of Address as received on 2/22/2007, and Amendment as received on 3/20/2007.

Information Disclosure Statement

3. It is desirable to avoid the submission of long lists of documents if it can be avoided. Eliminate clearly irrelevant and marginally pertinent cumulative information. If a long list is submitted, highlight those documents which have been specifically brought to applicant's attention and/or are known to be of most significance. See Penn Yan Boats, Inc. v. Sea Lark Boats, Inc., 359 F. Supp. 948, 175 USPQ 260 (S.D. Fla. 1972), aff'd, 479 F.2d 1338, 178 USPQ 577 (5th Cir. 1973), cert. denied, 414 U.S. 874 (1974). But cf. Molins PLC v. Textron Inc., 48 F.3d 1172, 33 USPQ2d 1823 (Fed. Cir. 1995). See MPEP 2004.

4. An applicant's duty of disclosure of material and information is not satisfied by presenting a patent examiner with "a mountain of largely irrelevant [material] from which he is presumed to have been able, with his expertise and with adequate time, to have found the critical [material]. It ignores the real world conditions under which examiners work." Rohm & Haas Co. v. Crystal Chemical Co., 722 F.2d 1556, 1573 [220 USPQ 289] (Fed. Cir. 1983), cert. denied, 469 U.S. 851 (1984). (Emphasis in original). Patent applicant has a duty not just to

disclose pertinent prior art references but to make a disclosure in such way as not to "bury" it within other disclosures of less relevant prior art; See Golden Valley Microwave Foods Inc. v. Weaver Popcorn Co. Inc., 24 USPQ2d 1801 (N.D. Ind. 1992); Molins PLC v. Textron Inc., 26 USPQ2d 1889, at 1899 (D.Del 1992); Penn Yan Boats, Inc. v. Sea Lark Boats, Inc. et al., 175 USPQ 260, at 272 (S.D. Fl. 1972).

5. It is impractical for the examiner to thoroughly review each reference, given the number of references cited. By initialing each of the cited references on the accompanying 1449 forms, the examiner is merely acknowledging the submission of the cited references and merely indicating that only a cursory review was made of the cited references.

Claim Objections

6. Claim 2 is objected to because "optional_token", as used (as a field of an instruction), is merely a label and has no grammatical meaning. Applicant should define "optional_token" just as "label#", ctx#, etc. were defined.

Claim Rejections - 35 USC § 102

7. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

> A person shall be entitled to a patent unless –
>
> (e) the invention was described in (1) an application for patent, published under section 122(b), by another filed in the United States before the invention by the applicant for patent or (2) a patent granted on an application for patent by another filed in the United States before the invention by the applicant for patent, except that an international application filed under the treaty defined in section 351(a) shall have the effects for purposes of this subsection of an application filed in the United States only if the international application designated the United States and was published under Article 21(2) of such treaty in the English language.

8. Claims 1, 7-9, and 13-15 are rejected under 35 U.S.C. 102(e) as being anticipated by Saulsbury et al., U.S. Patent No. 6,314,510 (herein referred to as Saulsbury).

9. Referring to claim 1, Saulsbury has taught a computer program product residing on a computer readable storage medium (Fig.1, component 102; column 2, lines 42-45) comprising instructions (Fig.4), including a context branch instruction (Fig.4 and column 4, lines 44-45; note the "branch on zero" (bz) instruction) that, when executed, causes a data processing apparatus to select another instruction in an instruction stream from one of a branch target instruction associated with a label specified by the context branch instruction (Fig.4; note that the "bz next1" instruction will branch to the target instruction associated with label "next1", which is the "btst wr1, db1" instruction) and an instruction following the context branch instruction ("st wr0, [addr2]") based on a comparison of a current executing context number to a context number specified by the context branch instruction. The bz instruction is a branch on zero instruction, which means that if the context number specified by the branch instruction (zero) matches a current executing context number (i.e., the value to be compared to zero), then a branch will occur. In this program, the "bz next1" instruction checks to see if the current executing context number (which may be interpreted as either dirty bit db0 or the flag which would be checked by the bz instruction) is equal to 0, the specified context number. See column 4, lines 39-45. It should be noted that the branch is a context branch and the numbers are context numbers as the branches and numbers are associated with executing contexts (i.e., environments). See the title, abstract, and column 2, lines 40-41.

b) retrieving the selected other instruction. As is known, with a condition branch, as shown in Fig.4, either the target instruction or the subsequent instruction will be retrieved.

10. Referring to claim 7, Saulsbury has taught a method of operating a processor comprising:

a) performing a comparison of a context number of an executing context to a context number specified by a context branch instruction. See the "bz next1" instruction of Fig.4 and column 4, lines 44-45. This instruction evaluates an executing context number (dirty bit db0 or the flag set in testing db0) to see if it equals the context number specified by the branch instruction, i.e., 0, since it is a "branch if 0" instruction. See column 4, lines 39-45. Note that the branch is a context branch and the numbers are context numbers as the branches and numbers are associated with contexts (i.e., executing environments). See the title, abstract, and column 2, lines 40-41.

b) selecting another instruction in an instruction stream from one of a branch target instruction associated with a label specified by the context branch instruction (Fig.4; note that the "bz next1" instruction will branch to the target instruction associated with label "next1", which is the "btst wr1, db1" instruction) and an instruction following the context branch instruction ("st wr0, [addr2]") based on a comparison (if the comparison yields that the numbers match, the target instruction will be selected; otherwise the following instruction will be selected).

c) retrieving the selected other instruction. As is known, with a condition branch, as shown in Fig.4, either the target instruction or the subsequent instruction will be retrieved.

11. Referring to claim 8, Saulsbury has taught a method as described in claim 7. Saulsbury has further taught selecting comprises selecting the branch target instruction if the executing context number matches the specified context number. As discussed above, and as shown in Fig.4 and column 4, lines 39-45, if the dirty bit db0 is evaluated (by way of flag) and determined to be equal to 0, which is the condition specified by the bz instruction, then a branch will occur.

12. Referring to claim 9, Saulsbury has taught a method as described in claim 7. Saulsbury has further taught that the context number has valid values of 0, 1, 2, or 3. Again, see the bz instruction of Fig.4. Since the bz instruction is "branch if 0", 0 is a valid specified context number.

13. Referring to claim 13, Saulsbury has taught a computer program product residing on a computer-readable storage medium (Fig.1, component 102; column 2, lines 42-45), for causing a processor that executes multiple contexts to perform a function, comprises instructions (Fig.4) causing the processor to:

a) perform a comparison of a context number of an executing context to a context number specified by a branch instruction. See the "bz next1" instruction of Fig.4 and column 4, lines 44-45. This instruction evaluates an executing context number (dirty bit db0 or the flag set in testing db0) to see if it equals the context number specified by the branch instruction, i.e., 0, since it is a "branch if 0" instruction. See column 4, lines 39-45. Note that the branch is a context branch and the numbers are context numbers as the branches and numbers are associated with contexts (i.e., executing environments). See the title, abstract, and column 2, lines 40-41.

b) select another instruction in an instruction stream from one of a branch target instruction associated with a label specified by the context branch instruction (Fig.4; note that the "bz next1" instruction will branch to the target instruction associated with label "next1", which is the "btst wr1, db1" instruction) and an instruction following the context branch instruction ("st wr0, [addr2]") based on a comparison (if the comparison yields that the numbers match, the target instruction will be selected; otherwise the following instruction will be selected).

c) retrieve the selected other instruction. As is known, with a condition branch, as shown in Fig.4, either the target instruction or the subsequent instruction will be retrieved.

14. Referring to claim 14, Saulsbury has taught a product as described in claim 13. Saulsbury has further taught that the context branch instruction causes the processor to select the branch target instruction if the executing context number matches the specified context number. As discussed above, and as shown in Fig.4 and column 4, lines 39-45, if the dirty bit db0, by way of a zero-flag, is evaluated and determined to be equal to 0, which is the condition specified by the bz instruction, then a branch will occur.

15. Referring to claim 15, Saulsbury has taught a product as described in claim 13. Saulsbury has further taught that the context number has valid values of 0, 1, 2, or 3. Again, see the bz instruction of Fig.4. Since the bz instruction is "branch if 0", 0 is a valid specified context number.

16. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

> A person shall be entitled to a patent unless –
>
> (b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.

17. Claims 1, 7, 10, and 13 are rejected under 35 U.S.C. 102(b) as being anticipated by Steven et al., "ALU Design and Processor Branch Architecture," *Microprocessing and Microprogramming*, 1993 (as cited by applicant and herein referred to as Steven).

18. Referring to claims 1, 7, and 13, Steven has taught a computer program product residing on a computer readable storage medium comprising instructions (this is inherent as instructions

must be stored somewhere), including a context branch instruction (see page 268, section 4.4, and note the Bcc instruction) that, when executed, causes a data processing apparatus to select another instruction in an instruction stream from one of a branch target instruction associated with a label specified by the context branch instruction (page 268; note that the Bcc instruction will branch to the target instruction associated with the label) and an instruction following the context branch instruction (the fall-through instruction, which inherent exists in conditional branching) based on a comparison of a current executing context number to a context number specified by the context branch instruction. The Bcc instruction compares a register value (specified context number) to a current executing context number (either another register value or immediate specified by the branch instruction). It should be noted that the branch is a context branch and the numbers are context numbers as the branches and numbers are associated with executing contexts (i.e., environments).

b) retrieving the selected other instruction. As is known, with a condition branch such as the Bcc instruction, either the target instruction or the subsequent instruction will be retrieved.

Claim Rejections - 35 USC § 103

19. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

20. Claims 2 and 4-6 are rejected under 35 U.S.C. 103(a) as being unpatentable over Saulsbury in view of Perets et al., U.S. Patent No. 6,564,316 (herein referred to as Perets).

21. Referring to claim 2, Saulsbury has taught a computer program product as described in claim 1. Saulsbury has further taught that the branch instruction is of the format br=ctx[ctx#, label#], wherein the label is a symbolic label corresponding to an address of the other instruction, and wherein ctx# is the context number, wherein the syntax "br=ctx" represents a branching operation based on ctx# matching the current context number provided by the data processing apparatus. For the bz instruction, the context number (ctx#) is 0 and the branch destination is next1 (label#)". Again, see Fig.4. Saulsbury has not taught that the instruction format includes an optional_token field. However, Perets has taught a branch instruction which includes an optional token. See Fig.6, step 600, and column 5, line 66, to column 6, line 5. The programmer-optional field (delay slot field) is used to specify the amount of delay slot instructions, where the use of delay slots for execution significantly speeds up the execution time of branch instructions. See column 5, lines 59-62. As a result, in order to speed up execution, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Saulsbury to include the optional_token field (delay slot field) of Perets.

22. Referring to claim 4, Saulsbury in view of Perets has taught a computer program product as described in claim 2. Saulsbury has further taught that the specified context number has valid values of 0, 1, 2, or 3. See the bz instruction of Fig.4. Since the bz instruction is "branch if 0", 0 is a valid specified context number.

23. Referring to claim 5, Saulsbury has taught a computer program product as described in claim 1. Saulsbury has not taught that the branch instruction has an optional token. However, Perets has taught a branch instruction which includes an optional token. See Fig.6, step 600, and column 5, line 66, to column 6, line 5. The programmer-optional field (delay slot field) is used

to specify the amount of delay slot instructions, where the use of delay slots for execution significantly speeds up the execution time of branch instructions. See column 5, lines 59-62. As a result, in order to speed up execution, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Saulsbury to include the option token (delay slot field) of Perets.

24. Referring to claim 6, Saulsbury in view of Perets has taught a computer program product as described in claim 5. Perets has further taught that the instruction has an optional token that causes a processor to execute a number of instructions corresponding to the value of the optional token following the branch instruction before performing a branch operation. See column 3, lines 3-7.

25. Claims 10-12 are rejected under 35 U.S.C. 103(a) as being unpatentable over Saulsbury.

26. Referring to claim 10, Saulsbury has taught a processor that can execute multiple contexts and that comprises:

a) a register stack. See Fig.1, component 110.

b) Saulsbury has taught an execution unit (Fig.1, component 108) coupled to the register stack and a program control store that stores a context branch instruction (Fig.1, component 102, and column 2, lines 42-45) that causes the processor to:

b1) perform a comparison of a context number of an executing context to a context number specified by the branch instruction. See the "bz next1" instruction of Fig.4 and column 4, lines 44-45. This instruction evaluates an executing context number (dirty bit db0) to see if it equals the context number specified by the branch instruction, i.e., 0,

since it is a "branch if 0" instruction. See column 4, lines 39-45. Note that the branch is a context branch and the numbers are context numbers as the branches and numbers are associated with contexts (i.e., executing environments). See the title, abstract, and column 2, lines 40-41.

b2) select another instruction in an instruction stream from one of a branch target instruction associated with a label specified by the context branch instruction (Fig.4; note that the "bz next1" instruction will branch to the target instruction associated with label "next1", which is the "btst wr1, db1" instruction) and an instruction following the context branch instruction ("st wr0, [addr2]") based on a comparison (if the comparison yields that the numbers match, the target instruction will be selected; otherwise the following instruction will be selected).

b3) retrieve the selected other instruction. As is known, with a condition branch, as shown in Fig.4, either the target instruction or the subsequent instruction will be retrieved.

c) Saulsbury has not explicitly taught a program counter for each executing context. However, this limitation is deemed to be inherent by the examiner. A program counter must exist because the program counter is the component that holds the address of the next instruction to be fetched. Without the PC, the system would not be able to fetch an instruction, which means that no work would be done. A program counter has to exist for each context because each context includes instructions that need to be fetched and executed.

d) Saulsbury has not taught that the execution unit is an arithmetic logic unit. However, Official Notice is taken that ALUs and their advantages are well known execution units that are expected

in the art. An ALU is the term given to a unit that is able to perform arithmetic operations (addition, subtraction, etc.) and logical operations (AND, OR, NOT, etc.). Clearly, by implementing an ALU, the system would be able to perform a variety of operations on data. As a result, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify the execution unit of Saulsbury to be an ALU so that Saulsbury is able to perform both arithmetic and logical operations.

27. Referring to claim 11, Saulsbury has taught a processor as described in claim 10. Saulsbury has further taught that the context branch instruction causes the processor to select the branch target instruction if the executing context number matches the specified context number. As discussed above, and as shown in Fig.4 and column 4, lines 39-45, if the dirty bit db0, or the flag set in testing db0, is evaluated and determined to indicate 0, which is the condition specified by the bz instruction, then a branch will occur.

28. Referring to claim 12, Saulsbury has taught a processor as described in claim 10. Saulsbury has further taught that the context number has valid values of 0, 1, 2, or 3. Again, see the bz instruction of Fig.4. Since the bz instruction is "branch if 0", 0 is a valid specified context number.

29. Claim 10 is rejected under 35 U.S.C. 103(a) as being unpatentable over Steven.

30. Referring to claim 10, Steven has taught a processor that can execute multiple contexts and that comprises:

a) a register stack. See page 268, and note that since registers are specified by instructions, a register stack exists.

b) an arithmetic logic unit (See the ALU of Fig.8) coupled to the register stack and a program control store that stores a context branch instruction (this is deemed inherent as a memory must exist to store instructions) that causes the processor to:

b1) perform a comparison of a context number of an executing context to a context number specified by the branch instruction. See section 4.4 on page 268. The Bcc instruction compares a register value (specified context number) to a current executing context number (either another register value or immediate specified by the branch instruction). It should be noted that the branch is a context branch and the numbers are context numbers as the branches and numbers are associated with executing contexts (i.e., environments).

b2) select another instruction in an instruction stream from one of a branch target instruction associated with a label specified by the context branch instruction (see page 268; note that the Bcc instruction will branch to the target instruction associated with the label when the comparison yields one result) and an instruction following the context branch instruction (the fall-through instruction, which inherently exists in conditional branching) based on the comparison (if the comparison yields that the numbers match, the target instruction will be selected; otherwise the following instruction will be selected).

b3) retrieve the selected other instruction. As is known, with a condition branch, either the target instruction or the subsequent instruction will be retrieved.

c) Steven has not explicitly taught a program counter for each executing context. However, this limitation is deemed to be inherent by the examiner. A program counter must exist because the

program counter is the component that holds the address of the next instruction to be fetched. Without the PC, the system would not be able to fetch an instruction, which means that no work would be done. A program counter has to exist for each context because each context includes instructions that need to be fetched and executed.

Response to Arguments

31. Applicant's arguments filed on March 20, 2007, have been fully considered but they are not persuasive.

32. Applicant argues the novelty/rejection of claim 1 on page 9 of the remarks, in substance that:

> "While Saulsbury does not specifically describe what operation are performed in the course of executing a 'branch on zero' instruction, conventionally, the execution of such an instruction requires that the executing microprocessor determine if the zero-flag of the ALU is set to 0. In the example illustrated in Saulsbury's Fig.4, the zero-flag will presumably be set as a result of the execution of the instruction preceding the bz instruction...Thus, unlike applicant's independent claim 1, a comparison operation is performed as a result of the instruction preceding Saulsbury's branch instruction, and not as a result of the execution of the branch instruction itself."

33. These arguments are not found persuasive for the following reasons:

a) A branch on zero instruction will check the zero flag. More specifically, the instruction will cause a comparison to occur between zero and the value of the zero-flag. If the zero-flag indicates 0, then the branch will occur. If the zero-flag indicates not-zero, then the branch will not occur. In either case, however, a comparison happens in response to the branch executing. The comparison is between the value specified by the branch and the flag controlling the branch. The preceding instruction merely sets the flag used in the comparison by the branch instruction.

34. Applicant argues the novelty/rejection of claim 1 on page 9 of the remarks, in substance that:

> "Moreover, such a comparison operation would compare the content of two registers and not the current executing context number, as provided by the microprocessor, to a context number value specified by the branch instruction."

35. These arguments are not found persuasive for the following reasons:

a) The examiner asserts that applicant is reading "current executing context number" too narrowly. As the examiner stated, any number in the machine can be considered a context number, as a context is essentially an execution environment, and all values produced by a microprocessor are done so within an execution environment. Therefore, the numbers compared are context numbers.

Conclusion

36. Applicant's amendment necessitated the new ground(s) of rejection presented in this Office action. Accordingly, **THIS ACTION IS MADE FINAL**. See MPEP § 706.07(a). Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the date of this final action.

Any inquiry concerning this communication or earlier communications from the examiner should be directed to David J. Huisman whose telephone number is (571) 272-4168. The examiner can normally be reached on Monday-Friday (8:00-4:30).

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Eddie Chan can be reached on (571) 272-4162. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

DJH
David J. Huisman
May 17, 2007